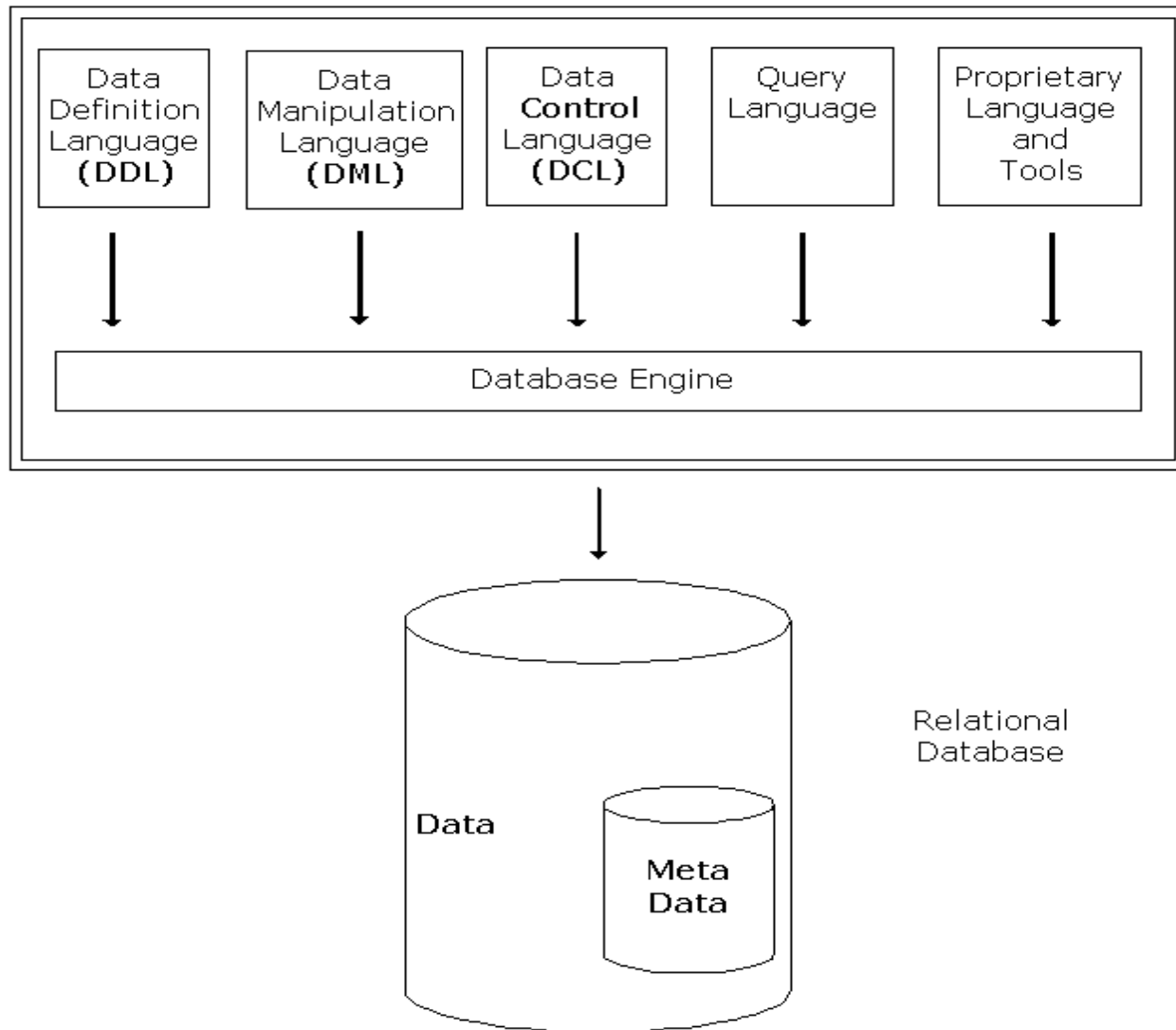


SDEV1201 Database Fundamentals

LESSON 10

Let's review

Database Management System (DBMS)



The database engine is the core set of programs that make up a DBMS. These programs execute when a request is submitted to the DBMS and directly interact with the database itself to provide a service. For example, a user adds new data to the database. The user would execute an application program to perform this task. The application program would submit a DML statement to the DBMS requesting that new data be added to the database. The DML statement causes one or more programs in the database engine to execute and actually add the new data to the database. It is important to note that developers and users do not interact directly with the database; instead they submit requests to the DBMS. The DBMS responds to requests from the various stakeholders in the system (e.g. users or developer) and performs the various tasks involved in creating, changing, and using a database.

The Relational DBMS

A relational DBMS uses tables to store data in the database. A table consists of rows and columns (similar to a spreadsheet). For example, information about customers would be stored in a table named Customer in the database as shown below:

<u>CustomerNumber</u>	FirstName	<u>LastName</u>	Address	Phone
100	Adams	Julie	9825-77 Ave Edmonton AB T5C 8E2	780-111-1111
200	Smith	Miles	4308-133 Ave Edmonton AB T5C 8E9	780-222-2222

Entity Relationship Model

The **Entity-Relationship** model is a tool that organizes and documents the logical design of a database. The entity-relationship model describes the data used by an application in terms of: **entities, attributes, and relationships.**

Entity Relationship Model

Entity

An entity is a person, place, thing or concept that is used in the business context of the application and for which data is collected.

Attributes

A piece of data that describes an entity is called an **attribute**. Synonyms for attribute include: element, property and field. In the order example, the customer entity has the following attributes:

- Customer Number
- Last Name
- First Name
- Address
- Phone Number

The values for all attributes describing an entity make up one instance of the entity. For example, the following values describe one specific customer.

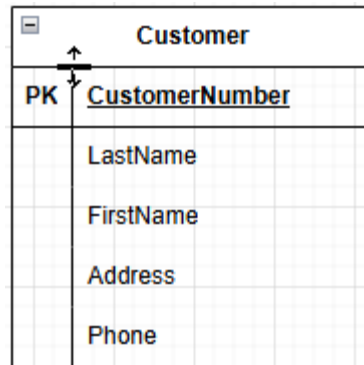
Customer Number	Last Name	First Name	Address	Phone
100	Adams	Joan	9825-77 Ave Edm Ab T5C 8E2	488-8712

Entity Relationship Model

The logical database design process uses the entity-relationship model to represent the business scenario as a collection of entities and relationships. The entity-relationship diagram (ERD) documents the logical design and becomes a starting point for the physical design process, which defines the structure of the database.

The **Entity-Relationship** model is a tool that organizes and documents the logical design of a database. The entity-relationship model describes the data used by an application in terms of: **entities, attributes, and relationships.**

Entities and Tables



After the ERD design is complete, when the database is created the entities become tables.

<u>CustomerNumber</u>	FirstName	<u>LastName</u>	Address	Phone
100	Adams	Julie	9825-77 Ave Edmonton AB T5C 8E2	780-111-1111
200	Smith	Miles	4308-133 Ave Edmonton AB T5C 8E9	780-222-2222

Today's Objective

By the end of this section you will be able to understand:

- ☐ Data types.
- ☐ Create table.
- ☐ Drop table.

Data Types

Attribute Data Types

In PostgreSQL, each **column/attribute**, local variable, expression, and parameter has a related **data type**. A data type is an attribute that specifies the type of data that the object can hold: integer data, character data, monetary data, date and time data, binary strings, and so on.

PostgreSQL supplies a set of system data types that define all the types of data that can be used with PostgreSQL.

What Are Data Types?

Unlike humans, a computer does not know the difference between "1234" and "abcd." A data type is a classification that dictates what a variable or object can hold in computer programming. Data types are an important factor in virtually all computer programming languages. Each data type requires different amounts of memory and has specific operations that can be performed over it.

Datatypes - Numeric

Numeric Types

Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
decimal	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
money	8 bytes	currency amount	-9223372036854775808 to +9223372036854775807

Datatypes - Numeric

Integer

An integer is a whole number (i.e. no decimal places). Integers can be positive, negative, or zero.

Decimal (precision, scale)

The precision of a decimal datatype is the total number of significant digits (i.e. not leading zeroes, commas or the decimal point) in the complete number, that is, the number of digits to both sides of the decimal point. The scale of a decimal datatype is the number of decimal digits in the fractional part (i.e. to the right of the decimal point). Decimal and numeric are equivalent datatypes.

Datatypes - Numeric

Money

The money datatype stores a currency amount with a fixed fractional precision which is determined by the `lc_monetary` parameter based on the database's Locale setting. Locale refers to an application respecting cultural preferences regarding alphabets, sorting, number formatting, etc. In North America, money output would be a dollar sign, at least one digit to the left of the decimal point, and two digits to the right of the decimal point. If the value is over 999, commas will delimit thousands/millions/billions, etc.

Datatypes - Character

char (n)

varchar (n)

If specified, the length n must be greater than zero and cannot exceed 10,485,760. If varchar is used without length specifier, the type accepts strings of any length. If char lacks a specifier, it is equivalent to char (1).

Values of type char are physically padded with spaces to the specified width n and are stored and displayed that way. However, trailing spaces are treated as semantically insignificant and disregarded when comparing two values of type char.

The char datatype is used to store character strings of fixed length while varchar is used to store strings of variable length. In CHAR, If the length of the string is less than set or fixed-length then it is padded with extra memory space. In VARCHAR, If the length of the string is less than the set or fixed-length then it will store as it is without padded with extra memory spaces.

Datatypes – Date/Time

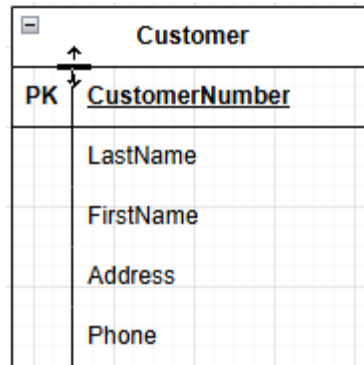
Name	Storage Size	Description	Low Value	High Value	Resolution
timestamp [(p)] [without time zone]	8 bytes	both date and time (no time zone)	4713 BC	294276 AD	1 microsecond
timestamp [(p)] with time zone	8 bytes	both date and time, with time zone	4713 BC	294276 AD	1 microsecond
date	4 bytes	date (no time of day)	4713 BC	5874897 AD	1 day

The time portion of all Date columns is automatically set to midnight (24-hour clock: 00:00:00, 12-hour clock: 12:00:00).

Possible code for creating tables for the Order example, are shown below.

Create Table

Entities and Tables



After the ERD design is complete, when the database is created the entities become tables.

<u>CustomerNumber</u>	FirstName	<u>LastName</u>	Address	Phone
100	Adams	Julie	9825-77 Ave Edmonton AB T5C 8E2	780-111-1111
200	Smith	Miles	4308-133 Ave Edmonton AB T5C 8E9	780-222-2222

Basic Table Definition

Invoices

Invoice Number	Invoice Date	Customer Number	Total
100	Jan 10, 2000	7	100.00
101	Jan 11, 2000	20	75.00

Table Name: Invoices

Attribute Name	Datatype
Invoice Number	Int (integer)
Invoice Date	Date
Customer Number	Int (integer)
Total	Decimal

Create Table

Most commercial DBMS use the relational model. Under the relational model, all data stored in the database is kept in tables. A table is like a two-dimensional array and consists of rows and columns. Each column records one attribute while each row records the values for all attributes that describe one instance of data. For example, we could use a table to record information about Invoices. If we needed to record the Invoice Number, Invoice Date, the Number of the Customer whom the invoice is for and the Total Amount Owing for the Invoice we would need four columns in the table. Each row in the table would record the information for one specific invoice.

Invoices Table:

Invoice Number	Invoice Date	Customer Number	Total
100	Jan 10, 2017	7	100.00
101	Jan 11, 2017	20	75.00

Basic table definition

9

- In SQL, we use a **CREATE TABLE** statement to create a table object in our database, which includes:
 - o The **name** of the table
 - o The **name** of each column
 - o The **data type** of each column
 - o Whether **NULL** is an **acceptable value** for the column
 - o Any other **constraints** that must hold true for a column

Create Table

Table names and column names should be descriptive. Limit the use of abbreviations in table or column names if possible. The bottom line is that a user of the database (e.g. a programmer) should be clear about what data is stored in a table or column based on its name. Some abbreviations are commonplace within industry (e.g. ID for Identification) and are acceptable. Experience will guide you as to what is an acceptable name. Many organizations have standards that define what an acceptable name is. If standards are in place they must be followed. If a name consists of multiple words, capitalize the first letter in each word.

CREATE TABLE **minimum syntax**

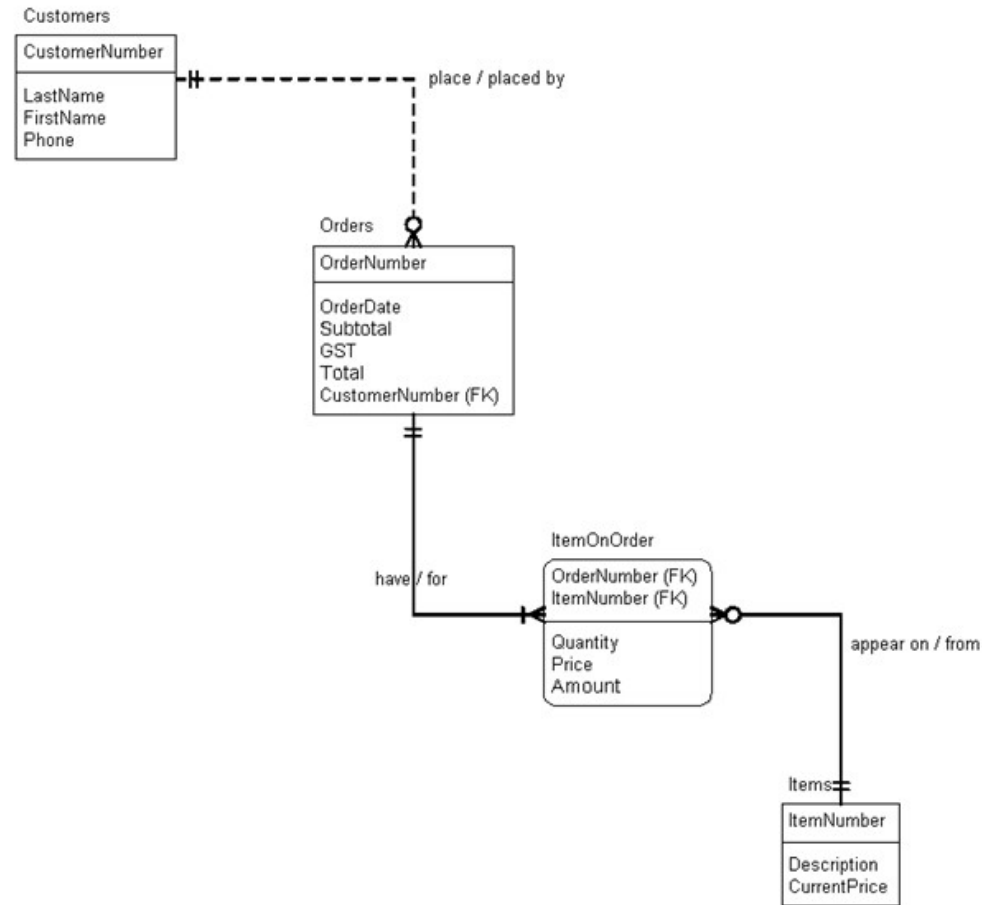
11

```
CREATE TABLE TableName (  
    Column1 DATATYPE  
    , Column2 DATATYPE  
    , Column3 DATATYPE  
    ...  
)
```

where *Column#* represents the name of the attribute/column and *DATATYPE* is *INT*, *DATETIME*, etc.

Orders example

12



Orders example

Create Table Customers

```
(  
    CustomerNumber integer not null,  
    LastName varchar (50) not null,  
    FirstName varchar (50) not null,  
    Phone char (10) null  
);
```

Create Table Orders

```
(  
    OrderNumber integer not null,  
    OrderDate date not null,  
    CustomerNumber integer not null,  
    Subtotal decimal (7,2) not null,  
    GST decimal (5,2) not null,  
    Total decimal (9,2) not null  
);
```

Orders example

Create Table Items

```
(  
    ItemNumber integer not null,  
    Description varchar (100) not null,  
    CurrentPrice decimal (6,2) not null  
);
```

Create Table ItemOnOrder

```
(  
    OrderNumber integer not null,  
    ItemNumber integer not null,  
    Quantity integer not null,  
    Price decimal (6,2) not null,  
    Amount decimal (7,2) not null  
);
```

Drop statement

18

The **DROP TABLE** statement is used to **delete** a table (both its definition and any data stored in it). The syntax is:

```
DROP TABLE TableName
```

e.g. to drop the **Items** table:

```
DROP TABLE Items
```

Homework

Do question 1 on the “Create Table Exercises PostgreSQL” exercise found in “Week 5 – Create Table” in Brightspace in the Activities section. The “Funky Flowers ERD” (located in the same location) will be needed for this exercise.

Reminder

Take-home Coding Assignment Advanced SELECT is due on Friday, October 10, 2025 at 5:00 PM.